

Visual Studio Add-in for Software Component Services in Smart Devices

MASTER THESIS OF SOFTWARE ENGINEERING (30 CREDITS ADVANCE LEVEL)

Hassan Aziz Khan

hkn10003@studnet.mdh.se

Irfan Ali

iai09001@student.mdh.se

Supervisor:

Frank Lüders

Frank.Luders@mdh.se

Examiner:

Sasikumar Punnekkat

sasikumar.punnekkat@mdh.se

School of Innovation, Design and Engineering

Västerås, Sweden

March 10, 2012

Dedications

We would like to dedicate this thesis to our beloved families and dear friends for their moral support and encouragement.

Abstract

In software world the term component based development is not unknown. In-fact in today's software industry we have a need to have such software development techniques which ensure quick and reliable products. This is where component based development(CBD) comes in. In CBD we develop reliable and reusable components which are used as building blocks of a software. But to successfully and easily integrate the components we ought to have some predefined standards and set of rules. These standards and set of rules make up a Component Model. It defines the rules for implementation, validation, integration and deployment of components. It also defines some runtime environment constraints such as interface of components and other runtime services. In Embedded Real Time Systems, software has a great significance as it has to respond in real time. So software built for a real time system must be very reliable and transparent. That's why normal software components cannot be used for the development of software for real time systems. A software for an embedded real time system should have some services like logging, execution time measurement and synchronization. Our thesis is about modifying a COM component so that it provides these services required for embedded real time systems. The concept of proxy objects can be used here to modify the already existent component. First the code of a simple COM component is regenerated and then these additional services are incorporated by using proxy objects.

Acknowledgement

We would like to thanks our supervisor Frank Lüders for his support, guidance and extensive support during the entire period of the thesis. His suggestions helped us in the development of the code generation tool as well as to complete thesis report.

Moreover we would like to thanks the school of Innovation, Design and Engineering at Malardalen University Sweden for providing us opportunity to work on this thesis work and helping us by providing the required resources, infrastructures to complete this thesis as well as our master degree successfully. The theoretical and practical course work during our master degree provided us the opportunity to learn new technologies software engineering techniques. It will definitely help us during our professional working and projects. We also would like to thanks our friends for their appreciation and suggestions during our thesis work.

At the end we would like to special thanks our families for their encouragement throughout our life to achieve the different milestones and their moral support during our thesis work.

Irfan Ali

Hasan Aziz Khan

Malardalen University

Västerås, Sweden.

Abstract	3
Acknowledgement.....	4
Contents	5
Chapter 1	7
1. Introduction.....	7
1.1 Problem Statement.....	8
1.2 Technology Adaptation and Scope.....	8
1.3 Report Outline	9
2. Component based software engineering	10
2.1 Software Component	11
2.2 Software Component Model.....	11
2.3 Software Component Specification	12
3. Software components and real time embedded systems.....	14
3.1 Real Time Systems	14
3.2 Embedded Real Time systems.....	14
3.3 Characteristics and Challenges of Real-Time Systems	15
3.4 Real-time Component Technologies	15
3.4.1 PECT	16
3.4.2 Koala	16
3.4.3 Rubus Component Model.....	16
3.4.4 Port Based Objects (PBO).....	16
3.4.5 PECOS	17
3.4.6 CORBA	17
3.5 Software Component Services for Embedded Real-Time Systems	17
3.5.1 Logging Service	18
3.5.2 Execution Time Measurement	19
3.5.3 Time Deterministic Synchronization	19
4. Visual Studio Add-in	20
4.1 Visual Studio	20
4.2 Visual studio Add-Ins.....	21
4.3 User Manual for Add-In	21
5. Code generation	28

5.1	Proxy Object	28
5.2	Code Generator	29
5.3	Test Application	29
6.	Conclusion and Future works	37
7.	References.....	38
	Appendix	40

Chapter 1

1. Introduction

Component based development (CBD) is rapidly emerging approach and it introduces the idea of the reusability of components. In component based development (CBD) already developed component are integrated and assemble to develop a system. By applying CBD approach we can improve quality, productivity and usability of the high quality software [1]. CBD development also reduced the time and increase the efficiency of the software applications. Component Based Software Engineering (CBSE) is quite successful in desktop, server-side software development.

“A software component can be deployed independently and is subject to composition by third party” [2]. The Software component having set of interfaces by using these interfaces a component interact with the other components. The interface of the components contains specification of its access point. There are several component technologies that are used for the component based development from the desktop to distributed information systems like Microsoft COM, DCOM, COM+ and .Net component model, CORBA component model form OMG and JavaBeans and Enterprise JavaBeans from SUN [3]

.

Component Object Model (COM) is developed in 1993 by Microsoft and it can be written in any programming language. A COM object hold the other COM object by using one of its containment technique and by using another technique called aggregation a COM outer object can describe the functions of its inner object. COM+ and DCOM also provide support to the COM technology. Microsoft .NET frame work is used to build and run different software projects and it provides support for different programming languages[2].

Component Based Software Engineering (CBSE) is not much successful in the development of embedded software systems due to the certain requirements of these systems like time critical, limited memory resources and CPU time. Embedded system development is a growing and complex field so there is a strong need of deployment of software component models in this field. By using component based development decrease the cost and time and make it easy to build complex software [4].

The research is on-going in building new component models by the different organizations for the development of the embedded real time systems. These component models include PECOS, SaveCCM, Robus and Koala component model. PECOS development is used mostly in the devices field development whereas Koala component model is mainly used in consumer electronics like microwave, TV etc. SaveCCM technology is used to build the embedded real time systems [2]. Robus component model is developed by Arcticus systems and it is used for the development of small embedded systems. So we use different component models for the development of the embedded systems according to their characteristics and requirements [5].

1.1 Problem Statement

This thesis is based on software components and component models. A simple software component model cannot be used for real-time embedded systems because of some limitations. The goal of this thesis work is to modify and extend the existing COM type library for the ATL smart device. We will implement a major service used by real-time system which is:

➤ Logging Service

To do the modification we will create a visual studio 2008 add-in which will read a COM type library and then it will generate a tree/hierarchy of the classes and functions of the library. Then user will select the desired classes and functions for which he wants to generate the modified code. The add-in will then generate the code for those classes and functions and create a new COM type library with all the three services. Then this extended COM type library will be tested with a smart device emulator.

1.2 Technology Adaptation and Scope

We discussed technology adoption with our project supervisor and decided to use Visual Studio 2008 professional edition for Visual Studio Add-in development. The Add-in then generates the proxy object for ATLSmartDevice COM dll. The .Net framework for smart device application development, debugging and deployment is supported by Visual Studio 2008 professional version. To test and evaluate the implementation of the thesis work we used Visual Studio 2008 Professional. By using Visual Studio.Net 2008 we developed ATL Smart Device COM component and client application to test the project. At first we developed Visual Studio add-in that can read metadata.NET assembly of the .NET smart device

component also ATL smart device COM type library. COM object's properties, methods and definitions are stored in type library in the form of interfaces and these interfaces can be accessed by the client application at the run time and invoke object's method.

The main objective of our thesis is to develop a Visual Studio add-in that select ATL smart device COM component and generate tree hierarchy of the classes and the functions of the COM library. The desired functions and classes are selected from the tree for which we want to generate the modified code. The add-in will then generate the code for those classes and functions and create a new COM type library with the logging service. Then this extended COM type library will be tested with a smart device emulator. We developed a smart device test application "Body Guard" that we will use to test our extended COM type library.

1.3 Report Outline

The chapter 2 of the thesis report includes the basic concept of Component Based Software Engineering (CBSE) by giving background detail CBSE and defining Software Component its specification including Software Component Model.

In the chapter 3 software components with real time embedded systems are discussed by explaining the basics of real time systems, characteristics and challenges of the real time systems. The last part of the chapter includes the real time component models that are currently in used.

The chapter 5 describe the software component services and smart device projects. Next chapter 6 consist of Visual Studio Add-in by proceeding with Visual studio detail and Visual studio Add-in followed by User Manual for Add-ins. The Chapter 7 and 8 contains the details of Code generation and test application part of the project respectively. The last part of the report is ended by giving the Conclusion and future work of the thesis.

Chapter 2

2. Component based software engineering

Component based software engineering (CBSE) is a young discipline so the basic concepts of it need to elaborate although many concepts have not been completely explained. Different authors have different understandings of different concepts of CBSE in different situations. The core idea of CBSE is to reuse the software components.

“The rapidly emerging approach called component-based development (CBD) reestablishes the idea of reuse and introduces new elements.” [1]

The reusability of software component plays very important role in the software development and makes it easier to reuse the already built components. To use the software component efficiently in the other projects the component must be well specified, easy to understand, easy to adopt, easy to deliver and deploy, easy to modify and replace. [1]

The purpose of the CBSE is to obtain efficient development with minimum development time and better quality products. Interfaces are the only way used for communication with component. It is very important that our component contains maximum information which users require for using the component. Software engineers spend lot of time for developing software components but sometimes users and developers are unable to use the component due to presence of lack of information.

Software components contain code which performs actions upon activation of a component. Components behavior is black-box, that's why components are accessed by using interfaces. Interfaces contain all information which a user requires for using a component. Specifications of interface and components are similar and useful for both developers and users of a component. Interface contains information about operation and context dependencies. But, Specification contains information about interface, context dependencies and operations; additional abstract information for developers is also enclosed in specification.

2.1 Software Component

A component can be defined as “A software component is a unit of a composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party”[1].

The component communicate with its environments through the interfaces therefor it is very important to clearly specify the interfaces of the software component and encapsulating the implementation in the component. To make the component independent it is very important to separate the implementation part from its interfaces.

According to another definition of software Component “A component is a reusable part of software, which is independently developed, and can be brought together with other component to build large units. It may be adapted but may not be modified”[6]. A component is a compiled code without a program source or design. There are two major types of the components, General Components and implementation components. General components includes the user interface widgets dropped onto a canvas or class frameworks, but implementation components includes any executable code, source code, code template or interface specification.

2.2 Software Component Model

“A component model defines a set of standards and conventions used by the component developer.” [1]

“A software component model defines what components are, how they can be constructed, how they can be composed or assembled and how they can be deployed as well as how we can reason about all these operations on components.”[7]

A software component model plays important role for the development of the component based applications by assembling components without modification of existing components. The reusability of software components in component based projects decreases development time, cost of the projects and increases the reliability, availability factors. The standard operation of software component model includes the construct the individual component and communicating, interacting the different components in a system [3].

There are different Standardized component model available includes OMG’s CORBA, Microsoft Component Object Model (COM) and Distributed COM (DCOM), COM+ including .Net component Model and sun’s JavaBeans and Enterprise JavaBeans [8].

Common Object Request Broker Architecture (CORBA) component model manages the details of components interoperability and allow the applications to communicate with each other through interfaces using Interface Definition Language (IDL) on different locations. The Objects Request Broker (ORB) which is important part of the CORBA is used to establish the client server relationships between components. The CORBA is used in component based software systems as well as in the Object oriented systems. [8]

The Component Object Model (COM) by Microsoft is widely used Component Model for the desktop and server side applications and it is more suitable for the embedded real time systems because of its simplicity. COM component model is now mostly being replaced by .Net Technology for the desktop and server side application domains. [3], [9]

Due to its automatic memory management and garbage collections features which are main barrier ensuring predictable timing, .Net is not so popular for embedded real time systems. A wide use of Microsoft .Net is for the client side development and for the server side development Enterprise JavaBeans (EJB) is a popular component model.

As component models defines standard for naming, interfacing and binding also sets of standard for run time services for applications domain that are known as software component services. These component models are not still widely used in the real time embedded systems because of not fulfilling the special requirements for the embedded real time systems like timing predictability and limited use of resources like memory and CPU. Due to the wide use of software applications for real time embedded systems research is on-going to full fill the special requirements for real time embedded systems and define new Component Models [3].

2.3 Software Component Specification

A component consists of some code that can be executed on certain platforms and for the access of the component it contains interface. The code of the component performs functionality when invoked with the component based applications where as the interface having everything a user need to know in order to deploy it. The behaviour of the component should be black box where the user doesn't need to know the inner structure of the component so interface need to have all the required information about the component its context dependences and its operations. So component specifications are the specification of its interface. The specifications of the component provide precise and complete information for the component user as well as for the developer of the component [1].

Syntactic specification of a commonly used for the practical software development and includes the specifications used with technologies such as COM, CORBA and Sun's

JavaBeans. These specifications are implemented using Interface Description Language (IDL) and programming language. For the Semantic specification of a component UML and Object Constraint Language (OCL) are used and it includes pre and post conditions and constraints to invoke component's operation also what type of parameter value a component operation accept. [3] To invoke the software component successfully we need more information than its functionality like its structural properties, how it can be composed with the other components its performance, capacity and family properties that specifies the relationships among the related components. A component's specifications cannot be expected to complete with respect to all properties as software component cannot be delivered with complete and sufficient specifications. [1]

Chapter 3

3. Software components and real time embedded systems

3.1 Real Time Systems

Real-time systems have a significant importance so they are used in many domains like embedded automotive electronics, Air traffic control system etc.

A real-time system can be defined as “A real-time system is a system that must satisfy explicit response-time constraints or risk severe consequences, including failure” [10]. Real time systems must meet the timing constraints requirements and must satisfy all conditions. As requirements of timeliness and meeting deadlines by finishing specified task in the certain time limit also more than one event may occur simultaneously in simultaneous processing. Along with these real-time systems must also meet with the predictability and dependability requirements. If the real-time systems fail to satisfy the requirements it may result in injury, loss of life or it may causes environmental damage. [1]

Standard component models and technologies can only be used to a limited degree because they cannot ensure to meet specified requirements for the real-time systems. To overcome this we can use different concept for system design also component composition reasoning and reusing components must be applied. Components Based Development (CBD) for the real-time systems is quite attractive and successful in certain domains.

3.2 Embedded Real Time systems

Embedded systems are used widely in the development of many applications and systems that uses the embedded computers. Major systems includes vehicle control system, medical control equipment and mobile phones. Embedded systems having a computer that interact directly with the external devices. There are certain specifications like safety, reliability, limited hardware capacity which they must meet. The concept of reusability of real-time components in the development of real-time systems applies when we use SBSE methodology. The development of the reusability components for the real time is expensive and complex than the non-reusable and non-real-time components [1]. It is very important for the real time components to meet the time constraints and must perform the tight deadlines by using the limited embedded system resources.

3.3 Characteristics and Challenges of Real-Time Systems

Time constraints is one of the important characteristics for the real-time systems and they used to interact with a physical system. The system is considered to be failed even it delivered the correct results but not meet the time requirement for the system. Deadline can be described as the maximum time within which the task must be complete its execution with respect to an even and it is usually time constraints for the real time task [1]. Real-time systems can be divided in to hard and soft real time systems depending on the results form a missed deadline. The criteria for the hard real-time system is that all task must meet the deadline on the other hand in case of soft real-time systems it is desirable to meet the deadline but it is not essential. In case of hard real time systems any late data is considered to be bad data. The development of reusable components for the real time systems is achieved by using the CBSE methodology but designing these reusable components is more complex. In real time applications it is essential for the components to meet timing constrains or deadlines and keep production costs down. Due to the limited resources for the embedded systems they task must be completed within tight deadlines. Some times when the systems are large and to complex many times it becomes impossible to predict with certainty when the certain event will occur also the order of their occurrence. Even in these cases real-time systems respond the to the event within a specified and a predictable time limits. Real-time software must also handle the software and hardware failure in a predictable manners [1].

3.4 Real-time Component Technologies

Component technologies that are Commonly used for the development of different domain component like COM and .Net from Microsoft, CORBA from OMG and JavaBeans from SUN. These technologies are used frequently in desktop and distributed enterprise applications. These technologies are used less frequently in the embedded real-time applications due to their characteristics of excessive processing, memory requirements and unpredictable timing [1], [11]. When the systems are developed for the specific platforms it becomes difficult to reuse them for a other platforms in embedded real time systems. The various methodologies used for the reusability of embedded system components depends upon the different parameters. These technologies includes PECT, Koala, Rubus Component Model, PBO, PECOS and CORBA [3].

3.4.1 *PECT*

“A Prediction-Enabled Component Technology (PECT) is a development infrastructure that incorporates a component technology, development tools and analysis techniques”[11]. PECT is still under research in the Software Engineering Institute (SEI) at Carnegie Mellon University. PECT development technology consist of development tools and analysis techniques. PECT is more portable and introducible because it is independent of underlying technology. It is very important to understand underlying mapping technologies to understand this model.

3.4.2 *Koala*

The Koala component technology is used for the development of software in consumer electronics and it is developed by Philips. It reduces cost of consumer electronic by using the cheap hardware. Thread sharing technique of koala make it very efficient and effective by keeping number of thread low which results the low usage of memory [3].

“The Koala components are units of independent design, development and reuse they can interact with the environment or other components through explicit interfaces only” [11].

The Koala components having no dependences on each other and their source code are fully visible for the developers. Due to their component internal visibility it allows companies to perform white box testing on them.

3.4.3 *Rubus Component Model*

Rubus component model is developed by the arcticus along with research community efforts and it is used mostly in the Volvo construction Equipment. According to its functionality it is divided into two parts red and blue that are used for hard real-time requirements and soft real-time requirements respectively. Rubus components model is mostly used in real-time requirement systems. The rubus components are easy to inspect and test because it contains the source code of the components. It is constructed on the top of the operating system so hard to break the bound and it is coupled with the operating system very tightly. Rubus is very desirable both for the hard and soft real time systems [3]. The main objective of the rubus Model is to make easier the reuse of the parts of the system [11].

3.4.4 *Port Based Objects (PBO)*

“Port Based Objects (PBO) was developed as a part of the Chimera RTOS projects at the Advanced Manipulators Laboratory at the Carnegie Mellon University” [11]. It forms a framework for the development of sensor based control systems along with the reconfigurable

robotics applications. It makes easier the reuse minimizing the communication and synchronization. PBO is treated as an object with various ports for the real time communication. PBO works on the component level as well as the system level. Due to the tight coupling with the real time operating systems the reuse of components in the new systems becomes difficult.[3], [11]

3.4.5 PECOS

PECOS (PErvasiveCOmponents Systems) is a developed as project between ABB Corporate Research Center and academia. “The main objective of PECOS projects is to enable the component based technology for embedded systems especially for field devices”[11]. PECOS even consider the non-functional requirements to be able to assess the properties thoroughly during the construction time. PECOS is incorporated with the Unified Modeling language (UML) for developing the system and it is easier to understand because of its focus on the non-functional properties. The Behavior of the PECOS is Black-box because it did not allow the access of the source code [3].

3.4.6 CORBA

The Common Object Request Broker Architecture (CORBA) is developed by the Object Management Group (OMG). It is used to write platform independent application by providing certain set of rules. It hides the low-level complexity and offers a platform independent interface. OMG introduces minimum CORBA to use within smaller resource constrained system. Another variation of CORBA is introduced by OMG known as real-time CORBA that is used to handle clock synchronization, bounded execution times etc. [3], [11]

There are several limitations with the traditional CORBA like no standard way for the deployment of object implementation, limited extension of functionality and no standard object lifecycle [11]. Another model introduced by the OMG is known as CORBA Component Model (CCM) and it covers the limitations of traditional CORBA.

CCM provides the better reuse for servers and provide a greater flexibility for dynamic configuration for CORBA applications [11].

3.5 Software Component Services for Embedded Real-Time Systems

The excessive usages of component models have been observed in software development for distribution information system and desktop application. Normally JavaBeans & ActiveX are normally used for the development of desktop application. On the other hand Enterprises JavaBeans and COM+ are very popular for the development of distribution of information

system. But on the other hand in Embedded Real Time System these models are not much popular due to special requirements. No doubt these components models play a vital role for rapid development of desktop application and distribution information system. A considerable research proof that these component models with some enhancements can also plays an important role for the development of Embedded Real Time System. One possible solution is to use a mainstream component model e.g. Component Object Model (COM) for this purpose. No doubt study shows the Component Object Model and its extended version Distribution Component Models are not incompatible with real time requirements. But COM is choose on the base of its simplicity and also previously used in industry.[12]

So we can say that basic models can b used for Embedded Real Time System development with extend of their functionalities with standardized services. These Standardized services normally known as Software Component Services. Logging, Synchronization and transaction control are the example of some basic software component services.

3.5.1 Logging Service

This Service not only allows the developer to calculate the timing of the application but also provide help for determining the execution path of the application. The best feature of this service is no need to recompile the component for doing such tasks. For adding a logging in interface just does some addition in application configuration file with an entry. An XML example shows the functionality of logging service.

```
<application>

...

<component name="myProject.C2">

<interface name="IC2">

<service type ="Logging"/>

</interface>

</component>

...

</application>
```

The above example shows when client tries to create class C2 then it will get a proxy which is used to write a log message in log file after and before each call passing on. [12]

3.5.2 Execution Time Measurement

This service is used to measure the exact execution time of a component. Time is measured in respect of best, worst and in average case. Online scheduler used this information for setting its strategy. The required result can be achieved just adds an entry in configuration file of application. For this purpose programming is not required in server or client side of the component.[12]

3.5.3 Time Deterministic Synchronization

This service provides a mechanism to that case where different components access the same data at a same time. Normally in this type of situation creates the unnecessary data blockage. So with the help of synchronization service a developer can easily overcome on the data blockage. This mechanism provides a set of different policies like e.g. mutual exclusion and read/ writes policy. Normally in mutual exclusion policy mechanism blocks all the queuing thread except one and entertains to all queuing threads one by one according to their priorities. However reader / writer policy is different as compare to mutual exclusion policy. Because in this mechanism many read threads can be execute at the same time on the same data. But write operation is treating as exclusive execution. This mechanism ensure the situation if a read operation is execute on a data the other write operation cannot update the same data until the completion of read operation. This mechanism also reduces the locking time between the threads. [12]

Chapter 4

4. Visual Studio Add-in

4.1 Visual Studio

Visual studio is a development environment for Windows and .Net platforms and it is developed by Microsoft. It can be used to develop various applications like console applications, Windows applications, Windows mobile applications, Windows Services, ASP.NET applications and web services by using C++, C#, VB.NET and J# etc. [13]. “Visual studio development tool provides a rich environment where you can rapidly create advanced applications and it contains set of productive and debugging features” [14].

In 1997 Microsoft introduced Visual studio that contains all environments like Visual C++, Visual basic in one application. There are various version of visual studio available in the market for the development of different applications and projects.

Visual studio 2008 was introduced by Microsoft in the end of 2007 and it supports different versions of .Net Frameworks. Microsoft has provided some extensibility features in the Visual studio by using them the developers can extend Visual studio. There are three common approaches that are used to extend the Visual studio are Add-ins, Macros and packages. Add-ins are used to access Visual Studio APIs for IDE to automate task like coding and deployment and Macros helps to automate frequently repeated task in VS [15].

There are some important features of the Visual Studio that are described below [14]:

Page design: By using this feature off the visual studio we can create attractive pages with drag-and-drop integrated web form designer without using HTML.

Automatic error detection: One of the important feature of the visual studio is to detect and reports error before even running the application which can save the time wastage for the error detection.

Debugging tools: Debugging tools of the Visual Studio help to watch code in action and it tacks the contents of the variables. Applications can be test by using the Built-in web server of the visual studio that works for debugging.

IntelliSence: This is very helpful tool which provides statement completion for recognized objects and automatically lists information like function parameters.

4.2 Visual studio Add-Ins

Add-Ins are used for the different purposes like to reuse the code within organization, to speed common task and to enforce rules, consistency among developers [16].

“An Add-in can be assumed as a simple DLL that is created by the user and Visual studio .Net loads it into its own process” [17]. Add-in access the different parts of the IDE while communicating with the Visual studio .NET IDE. These parts includes code editor, tool windows, solutions and projects also it respond the events that take place in IDE. ADD-in is compiled into a DLL and it can be distributed without the source code [17].

By using add-in development cost can be decreased also it can improve the efficiency of the developers. A projects Wizard known as Visual Studio .Net Add-in project contained by the Visual studio .Net and it is used to create an Add-in as a starting point. By running an Add-in project in Visual studio .Net a new process of Visual Studio .Net started in which the add-in will run and it allow to debug it within the Visual studio .Net process. Add-in can access Visual studio automation object DTE that help to control Visual studio .Net within the add-in [17].

Different task can be performed using Add-ins like add new tool windows that cover all the auxiliary windows in Visual studio .Net. Another feature of the Add-in can be described as by adding new pages to the options dialog box. Another feature of new menu items and one can also enable or disable them. Add-in can be developed by using different languages like C++, C# and VB.NET. Add-in can be distributed without the distribution of its source code [17].

4.3 User Manual for Add-In

As of now, you understand what a Visual Studio add-in actually is, so let us now have a look at our add-in and how it works. To make it simple for the users to understand the working of our add-in, we will show step by step screen shots and also describe those screen shots in detail. Before showing you the screen shots we would first like to explain that in order to launch this add-in you need to place the "CodeGeneratorAddIn - For Testing.AddIn" file in the "Addins" folder of Visual Studio. If you have default directory settings for Visual studio then go to "Documents" folder, then open "Visual Studio 2008" folder and then open "Addins" folder. Here you can copy and paste your ".AddIn" file and then edit it with notepad

or any text editor to set the path for your project's .dll file. After setting up the add-in file you can now compile the "CodeGeneratorAddIn" source code. If you have set up the ".AddIn" file correctly then at this point you can access the add-in (CodeGeneratorAddIn) in "Tools" menu of Visual Studio IDE.

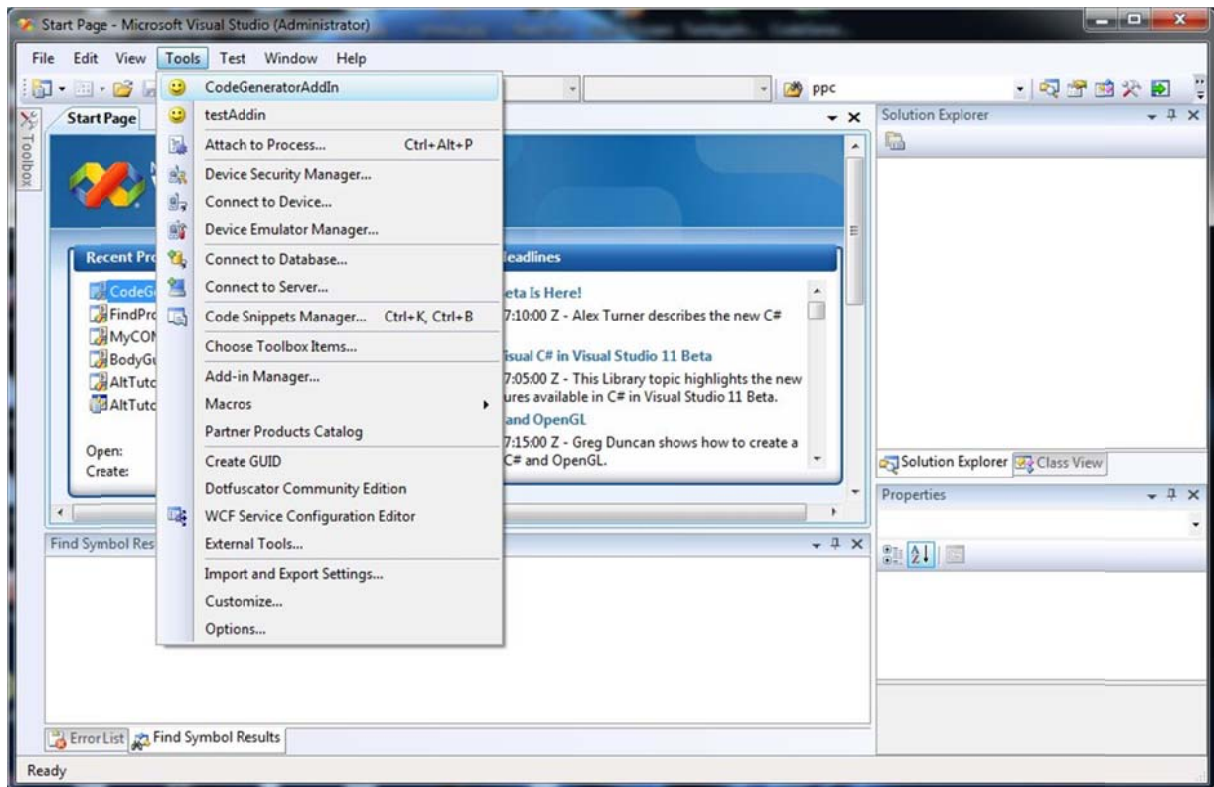


Figure 4.1: Add-in location

When you click on the CodeGeneratorAddIn a new window will open. This is the main window of Code Generator.

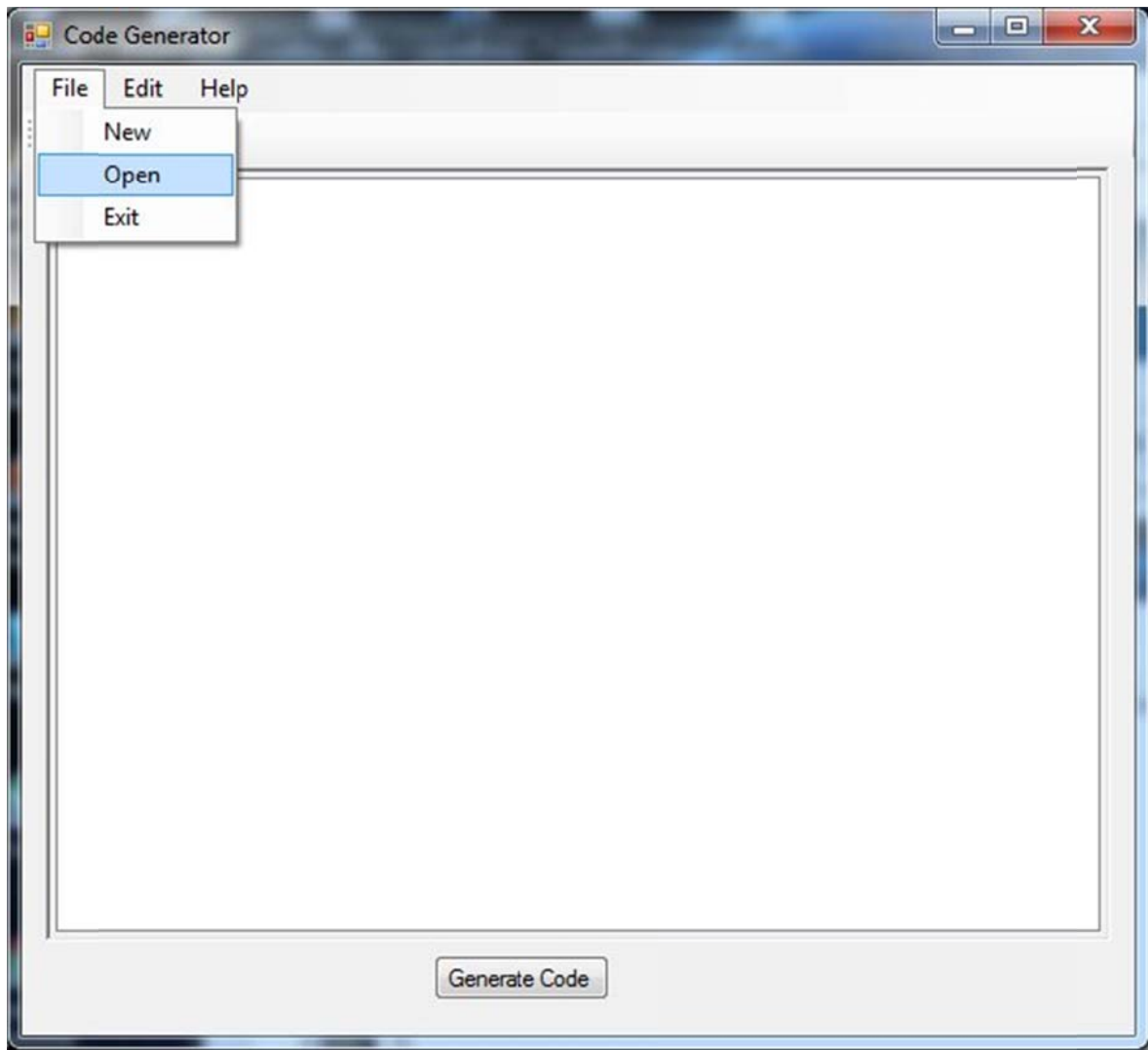


Figure 4.2: Code Generator Tool Main Window

Now you can open a .tlb, .dll or .exe file. To do that you need to go to "File" menu and then click on "Open". An open file dialog box will pop up through which you can select your desired library file.

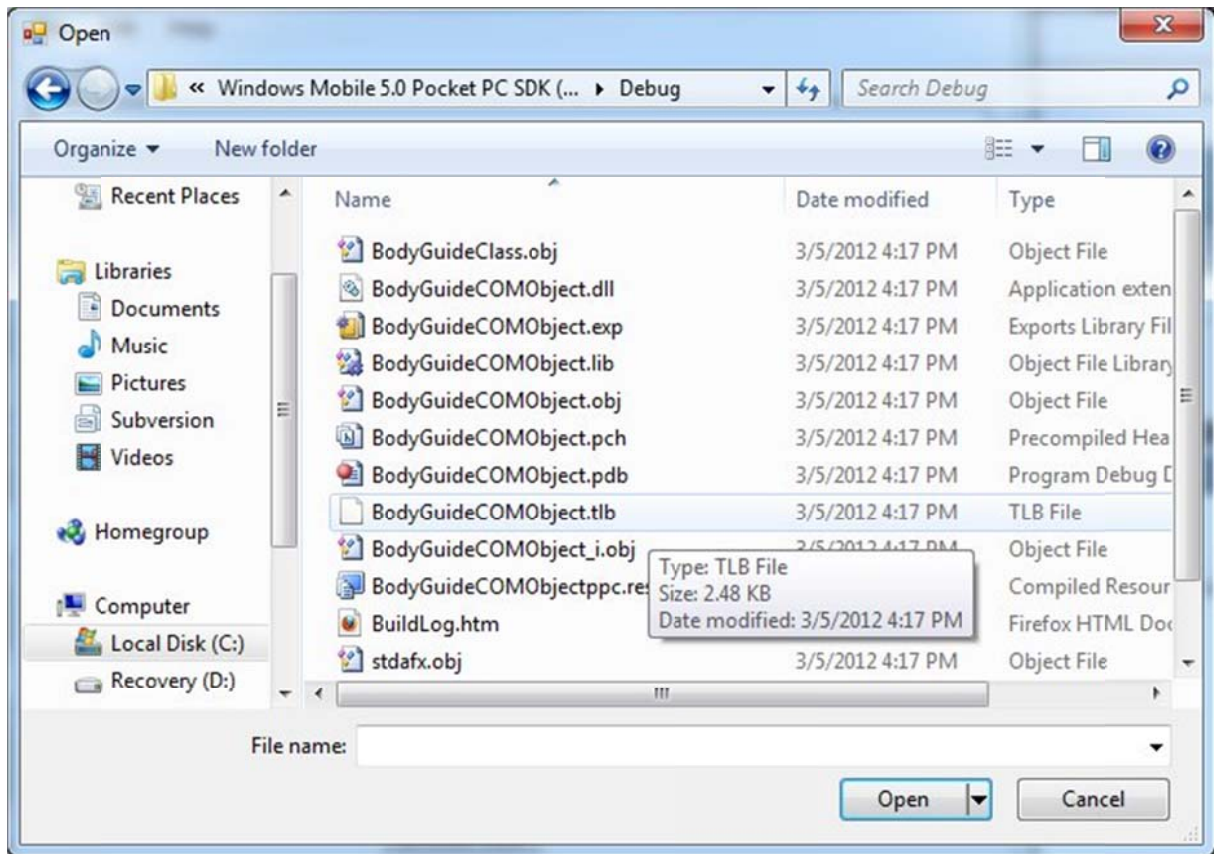


Figure 4.3: Browse and open a COM type library

Just select the library file and then click open. Code generator will then generate a class hierarchy for that library file. When you expand a specific class you can also see its Interfaces and methods. Now you can select the methods for which you want to add the logging service.

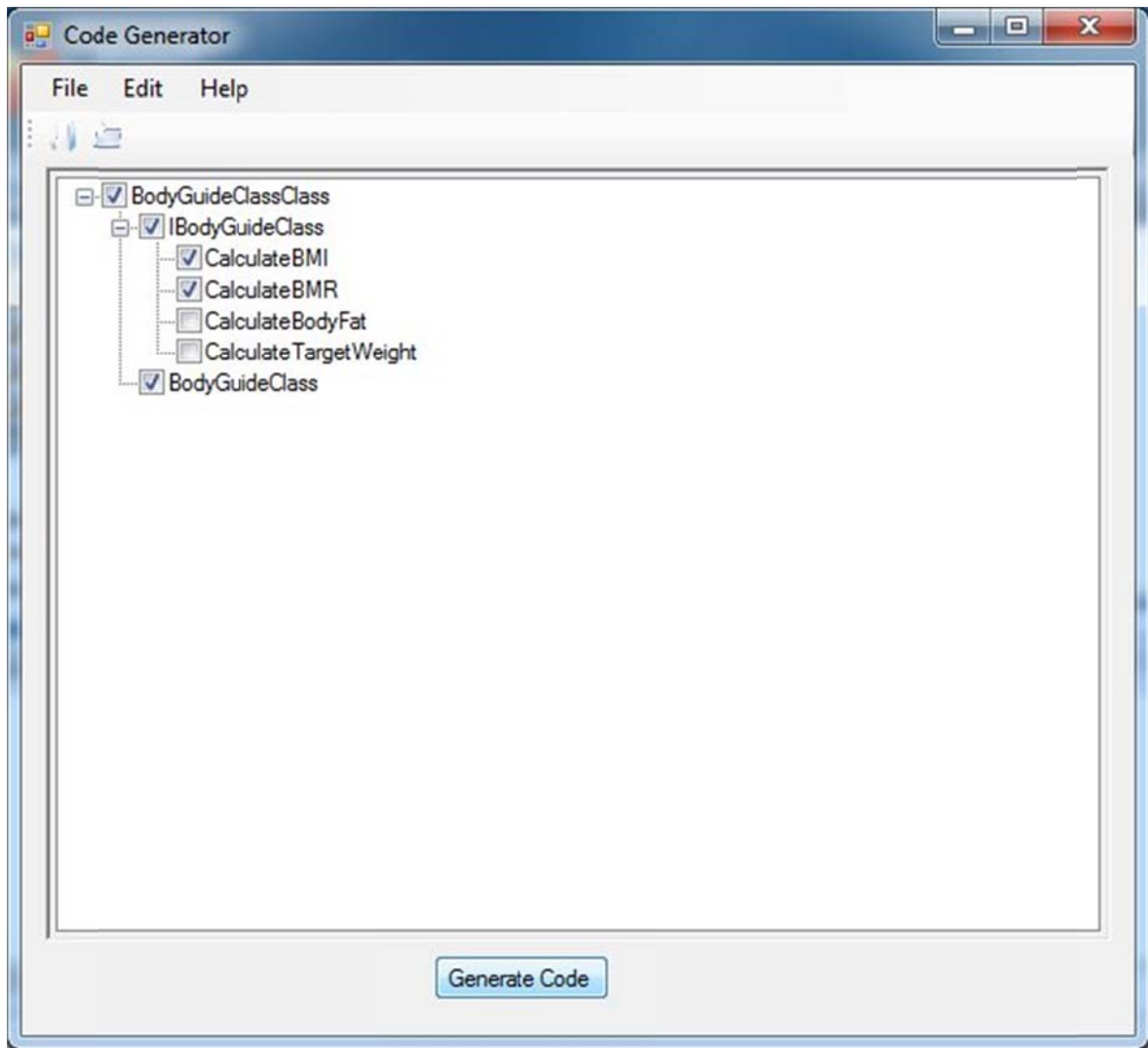


Figure 4.4: Select the methods for code generation

After selecting the desired methods you need to click on "generate Code" button. This will open a new dialog box which will ask you to select the visual c++ target project (Empty Smart Device Project). This is the project where your generated code and classes will be placed.

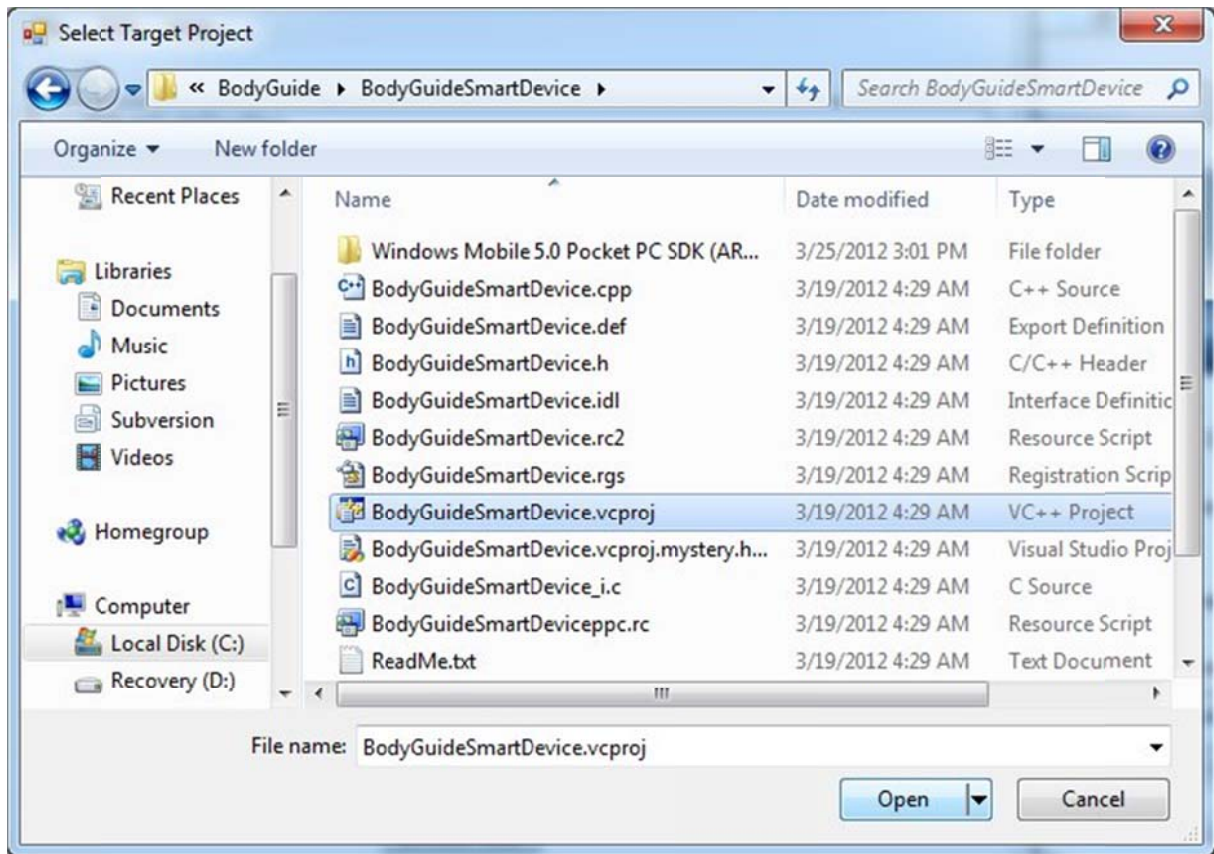


Figure 4.5: Select the target empty smart device project

Once you have select the target project, just click on open button to select it. Now our add-in will generate code and place it in the selected project. After placing the new files in the project, a new IDE window will open displaying the generated code. In the solution explorer you can see the newly generated proxy class(es), header file(s) and .idl file.

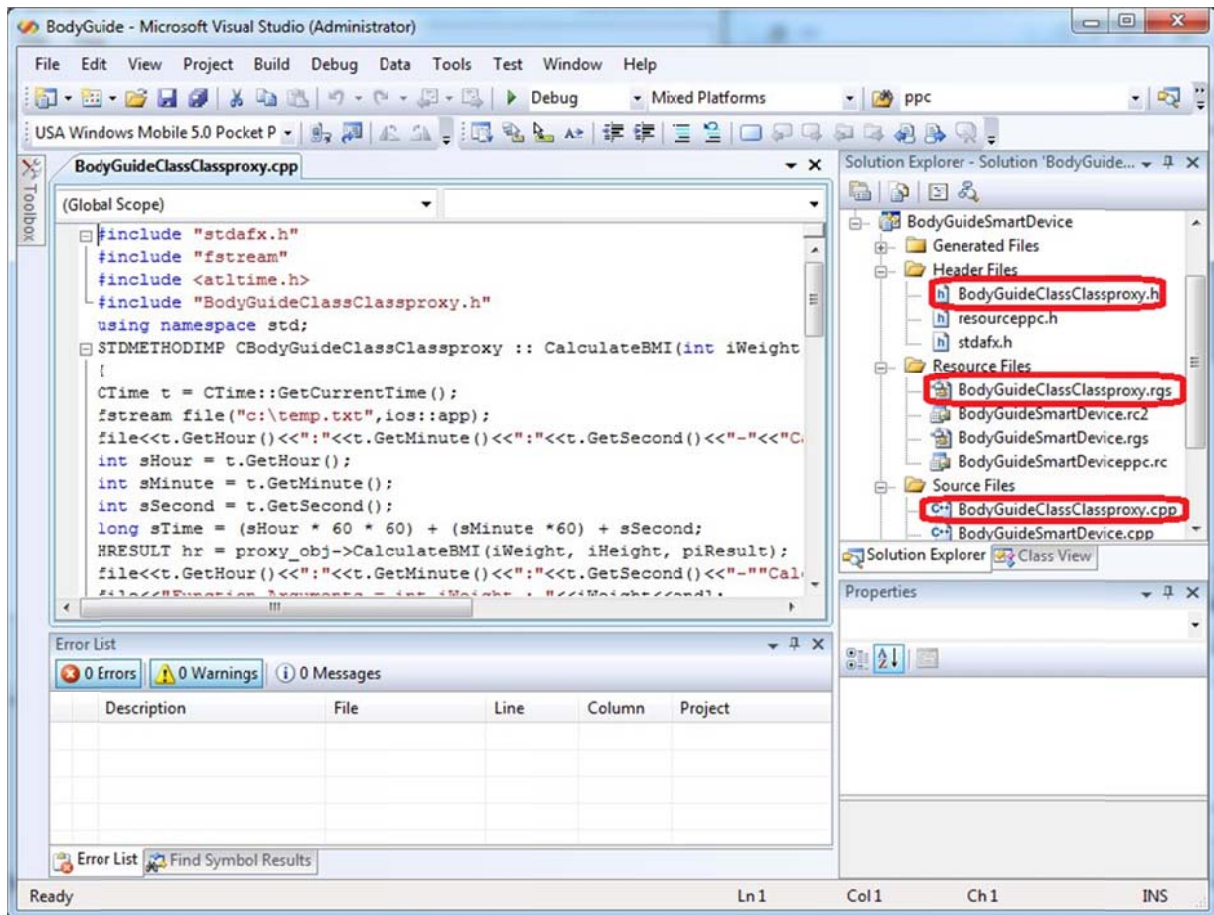


Figure 4.6: Generated Code

Chapter 5

5. Code generation

The concept of reusability of software component having a great importance in the development of embedded real time systems as well as smart device application development. Component Models define standards for naming, interface and the set of standard for the component services. The existing component model for the embedded real time systems cannot be used in all domain of real time due to their limitation for the specific domain. So in our thesis work we are going to extend a basic component model with the set of services which will cover large domain of embedded real time systems. We elaborate the code generator tool which supports software component services in smart device.

“The term smart device (SD) designates any physical object associated with computing resources and able to communicate with other similar objects via any physical transmission medium, and logical protocol, or with humans via standard user interface” [13]. Smart devices includes the big smart devices like PDAs and the small smart devices like RFID Tags.

Microsoft visual studio 2008 used to build applications for the various technologies and devices. Visual studio 2008 provide solutions that are based on the Windows Mobile 2003, Windows CE 5.0 and 6.0 also Windows mobile 5.0 and 6.0 for the smartphones and pocket PC.

Software component services are used with COM+ technologies to make it easier the development of software components for distributed information systems. The proxy objects are used to communicate and intercept the message calls between components to provide services like logging, time execution measurement, execution time and synchronization etc. Our thesis includes the implementation of the logging service for the real time systems. The main objective of the logging service is to trace the sequence of interaction between components also determining the execution time and path for the real time system.

5.1 Proxy Object

Proxy objects are used to perform the functionality of forwarding all calls and it's the public interface of the real class. The proxy object facilitate the components by intercept message call to provide services as logging, synchronization also execution time out services. Proxy object call the real objects it handle the call and when the call is done the it realises the real

object. Our focus here is on the logging service which trace out the sequence of objects interaction also it tell the execution time as well as path of the application. [6]

To implement logging services we can attach timing logic to each of the method call to monitor each method's execution time. This method seems more time consuming and it can create issues in the implementation of the component. Logging services can be implemented without the modification of the of the software components by creating through proxy object. For the implementation of the proxy we need to create a class which intercept all methods invocation. To implement proxy at first we need to define an interface also a class which will be used to implement this interface. For the implementation of all type of interfaces it work with proxy class that is generated at run time. For every class or interface .Net contain property "MethodInfo" that can be used to get method's name, type, it's return type also set of parameters passed into the invocation handler.[3]

5.2 Code Generator

The code generated tool is used to add logging service to the smart device .NET component and it uses the logic and concept of the proxy object. The code generator tool take as input the DLL file of the smart device component and it load it's assembly the read the modules and type information from DLL file. The DLL file contains the classes, methods and functions in it. By using System.Reflection library for each module of assembly it gives tree view control at parent root level for each class and it also populate methods and interface type at child root. By selecting the desired classes and methods the generator tool will generate code for C Sharp proxy classes and create Visual Studio .Net solution.

5.3 Test Application

In software development testing is a very important phase which ensures the quality and reliability of the software. Testers pay special attention to test the business logic of the software and make sure that all the requirements are met. Testing phase is equally import for component based software development and real time systems. As our thesis is on component based software development for smart devices so we had to pay more attention on testing phase as to make sure our add-in executes properly and flawlessly. Moreover we also had to make sure that all the requirements are met and the business logic is correct.

There are a lot of testing techniques being used in software industry. But the traditional and most common techniques are white box testing and black box testing. In white box testing we make sure that every statement of the code gets executed at least once. While in black box

testing we create scenarios to test the business logic and functional requirements. We also adopted these two techniques for testing our add-in. For white box testing we used the visual studio debugger which is a great way to ensure and inspect the code line by line. So this will enabled us to verify that application does not crashes at any statement and it has no abnormal behaviour. We developed such inputs which cause the program control to traverse through every statement in the code. Initially there were some unhandled exceptions in the code which were pointed out by the debugger and we fixed those bugs. We continued white box testing until all the bugs were fixed.

Then we moved on to black box testing. It is very important to verify that the software meets all the requirements and the business logic is correct. So we developed test cases for every requirement and carefully executed those test cases recoding every output. We carried out black box testing by developing a test application. First we created a COM component with name BodyGuideCOMObject. This component provided the functionality of a complete fitness advisor. It calculates your BMI (Body Mass Index), your BMR (Body Mass Ratio), your total body fat and then it advises you whether to lose or gain weight. It also tells you how much daily calories you need to take in order to gain or lose weight. Then we created a smart device client which used this COM component and provided the GUI for it. We can show you the screen shots of the client so that you can better understand it's working. The first screen is the main form which has 3 different options, BMI, BMR and Body Fat.

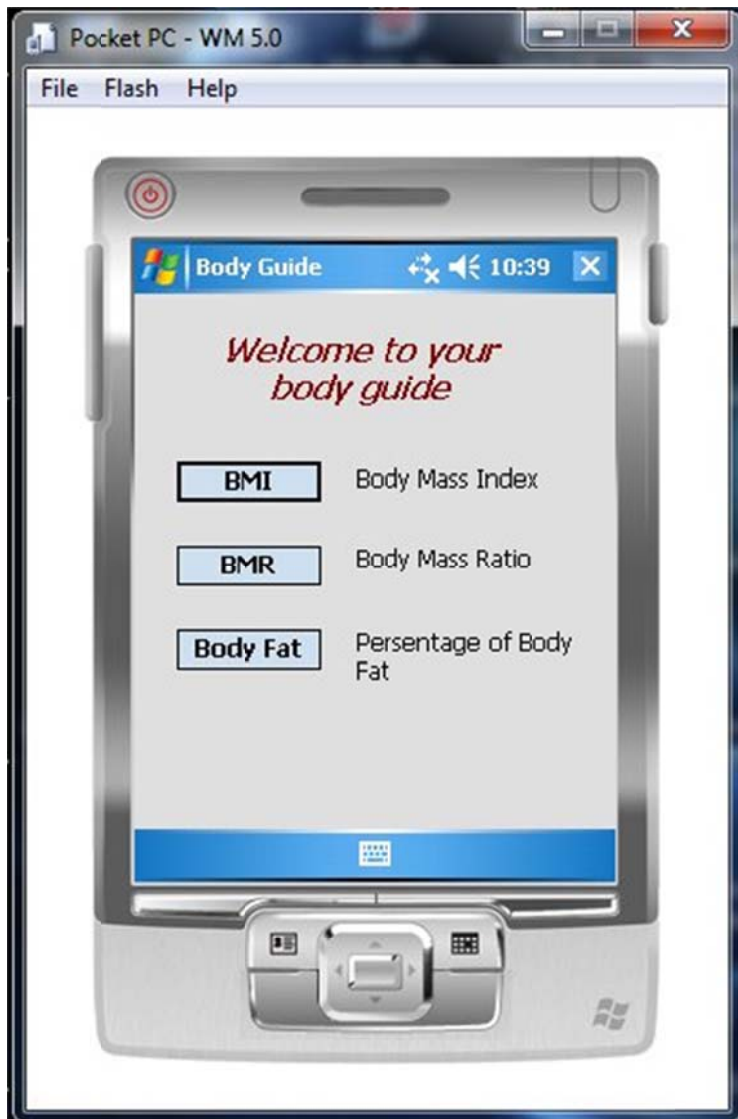


Figure 5.1: Body guide client main form

As you can see in the above screen shot you can click on any button to calculate the respective parameter. To get an advice you need to calculate all three parameters first. To calculate your BMI, click on BMI button. The BMI form will open where you will be asked to enter your height and weight.



Figure 5.2: Calculate BMI form

After providing the required details you can calculate the BMI by clicking on Calculate BMI button. To calculate your BMR you need to click on BMR button on main form. Then a BMR form will open which will ask you to enter height, weight, age and gender.



Figure 5.3: Calculate BMR form

After providing the required details you can calculate your BMR by clicking on the Calculate BMR button. This BMR is actually required to calculate your daily calories requirement. To calculate your total body fat you need to click on Body Fat button on the main form. Then a Body Fat form will open where you will need to enter your weight, waist and gender. For females, wrist, hips and forearm's measurement is also required.

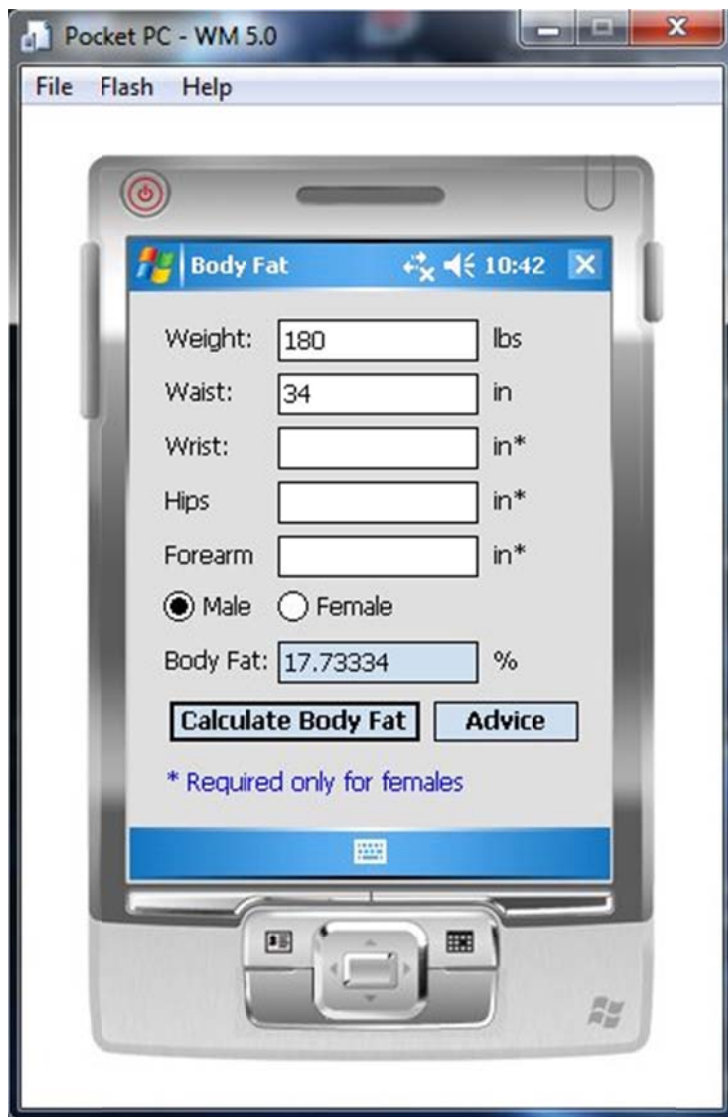


Figure 5.4: Calculate body fat form

After providing the required information you can calculate the body fat by clicking on Calculate Body Fat button. Once you have calculated the BMI, BMR and Body Fat you can proceed with getting an advice based on your results. To get an advice you need to click on Advice button on Body Fat form. Then you will have to enter a rough estimate of your daily activity.

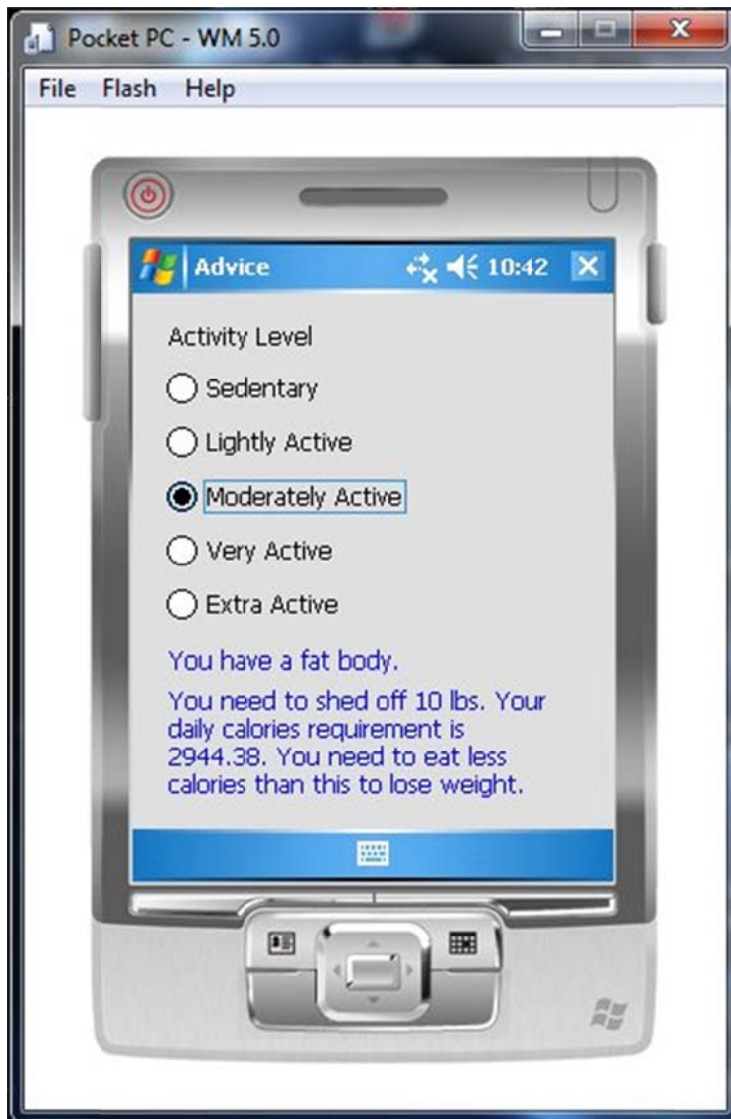


Figure 5.5: Advice form

The advice will tell you if you have a fat body or a lean body. It will also tell you how much weight you should lose or gain based on the calculations, and to lose or gain weight how much calories you should eat every day.

Once we have tested our COM component with the smart device client, we generated the code for the very same COM component with our GodeGenerator add-in. Then we used the same smart device client application with our generated proxy component. We executed all the test cases on it and recorded the outputs. There was no difference in the results. Now we will show you the architecture of our test application so you can understand the logic.

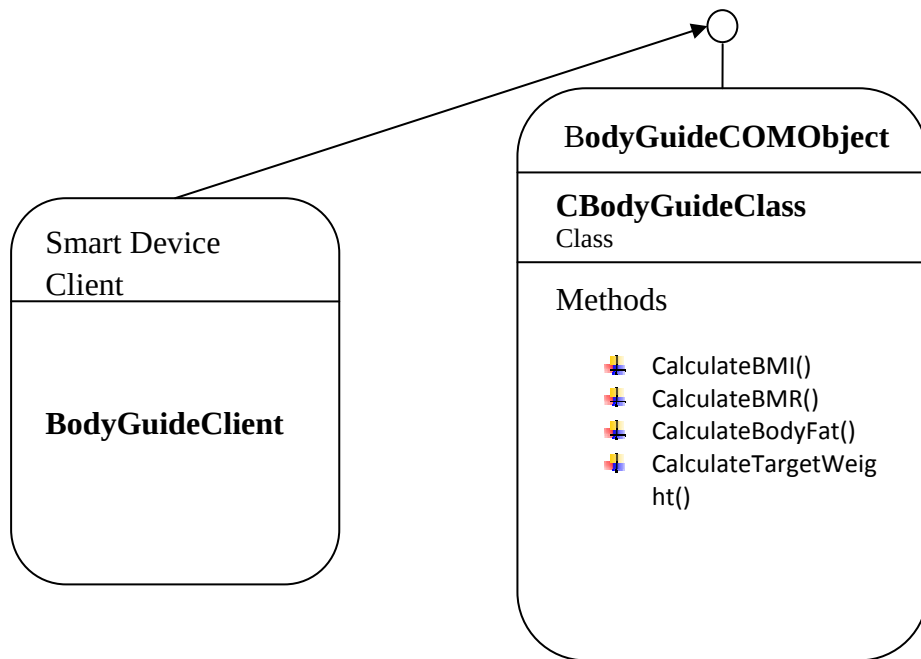


Figure 5.6: BodyGuideCOMObject directly interacting with SmartDeviceClient

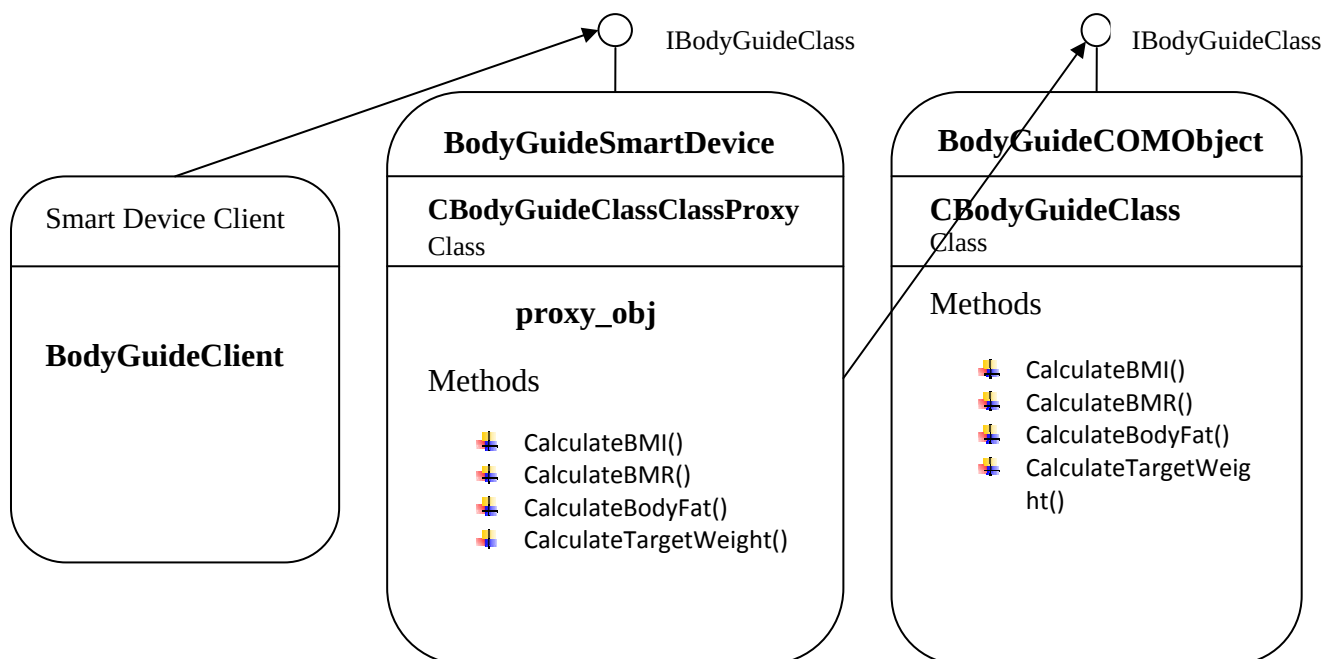


Figure 5.7: Generated Proxy Class placed in between COM object and Client

Chapter 6

6. Conclusion and Future works

Component Based Software development (CBSE) approach make it easier to develop the complex software by using the software components. The reusability of software component plays very important role in the software development. To reuse the software components efficiently in the other component based software projects the component must be well specified, easy to understand, easy to adopt, easy to deliver and deploy, easy to modify and replace. Component model plays important role by defining the set of standard for the development, implementation, composition and deployment of the software components. There are different component models that are available like OMG's CORBA, Microsoft Component Object Model (COM) and Distributed COM (DCOM), COM+ including .Net component Model and sun's JavaBeans and Enterprise JavaBeans. These component models are less frequently used for the development of embedded real time systems due to their certain limitations like excessive processing, memory requirements and unpredictable timing.

In our thesis work we have extended a basic component model with a set of services to cover the large application domain for embedded real time systems. We have extended the COM type model for the ATL smart device by adding the logging service into it. To accomplish this we created Visual Studio Add-in that read a COM type library and then it generates a tree hierarchy of classes and functions of the library. We can select our desired classes and functions for which we want to generate the modified code. The add-in extract the classes and functions and create a new COM type library and generates proxy object that adds logging service for smart device and extend the functionality of COM type library. To test the extended COM type library we created a test application "Body Guard" that verify the functionality of new COM type Library.

As our visual studio add-in read a COM type library and extend it only with the logging service but it does not provides component services like synchronization and execution time measurement can be implemented. In the future work remaining component services synchronization and execution time measurement can be implemented with the COM type library for the ATL smart device.

7. References

- [1] I. Crnkovic and M. P. H. Larsson, *Building reliable component-based software systems*. Artech House, 2002.
- [2] M. Zaheer, "Evaluation of Software Components in Embedded Real-Time Systems," 2007.
- [3] M. Ahmad, "A Code Generator for Software Component Services in Smart Devices," 2010.
- [4] F. Luders, S. Ahmad, F. Khizer, and G. Singh-Dhillon, "Using Software Component Models and Services in Embedded Real-Time Systems," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 2007, p. 286c–286c.
- [5] I. Afzal, "A Comparative Study of Software Component Models for Embedded Real Time Systems," 2007.
- [6] S. U. Rehman, "A Visual Studio Add-In for Software Component Services in Smart Devices," Mälardalen University, Sweden, 2011.
- [7] K. K. Lau and Z. Wang, "Software component models," *Software Engineering, IEEE Transactions on*, vol. 33, no. 10, pp. 709–724, 2007.
- [8] X. Cai, M. R. Lyu, K. F. Wong, and R. Ko, "Component-based software engineering: technologies, development frameworks, and quality assurance schemes," in *Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific*, 2000, pp. 372–379.
- [9] J. Q. Ning, "Component-based software engineering (CBSE)," in *Assessment of Software Tools and Technologies, 1997., Proceedings Fifth International Symposium on*, 1997, pp. 34–43.
- [10] F. Lüders, *An evolutionary approach to software components in embedded real-time systems*. Department of Computer Science and Electronics, Mälardalen University, 2006.
- [11] M. Akerholm and J. Fredriksson, "A Sample of Component Technologies for Embedded Systems," *Malardalen Real-Time Research Centre Technical Report*, 2004.
- [12] F. Luders, D. Flemstrom, and A. Wall, "Software Component Services for Embedded Real-Time.pdf." Mälardalen University, 2005.

- [13] C. Carabelea and O. Boissier, “Multi-agent platforms on smart devices: Dream or reality,” in *Proceedings of the Smart Objects Conference (SOC03), Grenoble, France*, 2003, pp. 126–129.
- [14] M. MacDonald, *Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional*, 2nd ed. Apress, 2007.
- [15] K. Nayyeri, *Professional Visual Studio Extensibility*. Wrox, 2008.
- [16] L. Smith, *Writing Add-ins for Visual Studio .NET*. Apress, 2002.
- [17] J. Cogswell, *Developing Visual Studio .NET Macros and Add-Ins*, 1st ed. Wiley, 2003.

Appendix A: BodyGuideCOMObjectCode

A.1: CPP File

```
// BodyGuideClass.cpp : Implementation of CBodyGuideClass

#include "stdafx.h"
#include "BodyGuideClass.h"

// CBodyGuideClass

STDMETHODIMP CBodyGuideClass::CalculateBMI(int iWeight, int iHeight,
double* pfBMI)
{
    // TODO: Add your implementation code here
    if(iHeight!=0)
    {
        *pfBMI = ((1.0 * iWeight)/(iHeight * iHeight))*703;
    }
    else
    {
        *pfBMI=0;
        return E_FAIL;
    }
    return S_OK;
}

STDMETHODIMP CBodyGuideClass::CalculateBMR(int iWeight, int iHeight, int
iAge, int iGender, float* pfBMR)
{
    // TODO: Add your implementation code here
    if(iGender == 1)
    {
        *pfBMR = 66 + ( 6.23 * iWeight ) + ( 12.7 * iHeight ) - ( 6.8 *
iAge );
    }
    else if(iGender==2)
    {
        *pfBMR = 655 + (4.35 * iWeight) + ( 4.7 * iHeight ) - ( 4.7 *
iAge );
    }
    return S_OK;
}

STDMETHODIMP CBodyGuideClass::CalculateBodyFat(int iWeight, int iWaist, int
iWrist, int iHips, int iForearm, int iGender, float* pfBodyFat)
{
    // TODO: Add your implementation code here
    if(iGender == 1)
    {
        float factor1 = (iWeight * 1.082) + 94.42;
        float factor2 = iWaist * 4.15;
        float leanMass = factor1 - factor2;
        float fat = iWeight - leanMass;
        *pfBodyFat = (fat * 100)/iWeight;
    }
}
```

```

    }
    else if(iGender == 2)
    {
        float factor1 = (iWeight * 0.732) + 8.987;
        float factor2 = iWrist / 3.140;
        float factor3 = iWaist * 0.157;
        float factor4 = iHips * 0.249;
        float factor5 = iForearm * 0.434;
        float leanMass = factor1 + factor2 - factor3 - factor4 +
factor5;
        float fat = iWeight - leanMass;
        *pfBodyFat = (fat * 100)/iWeight;
    }
    return S_OK;
}

STDMETHODIMP CBodyGuideClass::CalculateTargetWeight(int iWeight, float
fBodyFat, int iGender, int* iTargetWeight)
{
    // TODO: Add your implementation code here

    int fatMass = 0;
    int leanBodyMass=0;
    fatMass = iWeight * (fBodyFat / 100);
    leanBodyMass = iWeight - fatMass;
    if(iGender==1)
        *iTargetWeight = leanBodyMass + (iWeight * 0.12);
    else
        *iTargetWeight = leanBodyMass + (iWeight * 0.19);
    return S_OK;
}

```

A2: Header File

```

// BodyGuideClass.h : Declaration of the CBodyGuideClass

#pragma once
#ifdef STANDARDSHELL_UI_MODEL
#include "resource.h"
#endif
#ifdef POCKETPC2003_UI_MODEL
#include "resourceppc.h"
#endif
#ifdef SMARTPHONE2003_UI_MODEL
#include "resourcesp.h"
#endif
#ifdef AYGSHELL_UI_MODEL
#include "resourceayg.h"
#endif

#include "BodyGuideCOMObject.h"

// CBodyGuideClass

```

```

class ATL_NO_VTABLE CBodyGuideClass :
    public CComObjectRootEx<CComMultiThreadModel>,
    public CComCoClass<CBodyGuideClass, &CLSID_BodyGuideClass>,
    public IBodyGuideClass
{
public:
    CBodyGuideClass()
    {
    }

#ifdef _CE_DCOM
    DECLARE_REGISTRY_RESOURCEID(IDR_BODYGUIDECLASS)
#endif

    BEGIN_COM_MAP(CBodyGuideClass)
        COM_INTERFACE_ENTRY(IBodyGuideClass)
    END_COM_MAP()

    DECLARE_PROTECT_FINAL_CONSTRUCT()

    HRESULT FinalConstruct()
    {
        return S_OK;
    }

    void FinalRelease()
    {
    }

public:
    STDMETHODCALLTYPE(CalculateBMI)(int iWeight, int iHeight, double* pfBMI);
    STDMETHODCALLTYPE(CalculateBMR)(int iWeight, int iHeight, int iAge, int
iGender, float* pfBMR);
    STDMETHODCALLTYPE(CalculateBodyFat)(int iWeight, int iWaist, int iWrist, int
iHips, int iForearm, int iGender, float* pfBodyFat);
    STDMETHODCALLTYPE(CalculateTargetWeight)(int iWeight, float fBodyFat, int
iGender, int* iTargetWeight);
};

OBJECT_ENTRY_AUTO(__uuidof(BodyGuideClass), CBodyGuideClass)

```

A3: IDL file

```

// BodyGuideCOMObject.idl : IDL source for BodyGuideCOMObject
//

// This file will be processed by the MIDL tool to
// produce the type library (BodyGuideCOMObject.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";

[
    object,
    uuid(EDE969FE-E052-4880-AEDB-51F57260CDF9),
    helpstring("IBodyGuideClass Interface"),

```

```

        pointer_default(unique)
    ]
interface IBodyGuideClass : IUnknown{
    [helpstring("method CalculateBMI")] HRESULT CalculateBMI([in] int
iWeight, [in] int iHeight, [out,retval] double* pfBMI);
    [helpstring("method CalculateBMR")] HRESULT CalculateBMR([in] int
iWeight, [in] int iHeight, [in] int iAge, [in] int iGender, [out,retval]
float* pfBMR);
    [helpstring("method CalculateBodyFat")] HRESULT CalculateBodyFat([in]
int iWeight, [in] int iWaist, [in] int iWrist, [in] int iHips, [in] int
iForearm, [in] int iGender, [out,retval] float* pfBodyFat);
    [helpstring("method CalculateTargetWeight")] HRESULT
CalculateTargetWeight([in] int iWeight, [in] float fBodyFat, [in] int
iGender, [out,retval] int* iTargetWeight);
};
[
    uuid(BA777CDD-5744-4566-AFD8-07ABE8E7029A),
    version(1.0),
    helpstring("BodyGuideCOMObject 1.0 Type Library")
]
library BodyGuideCOMObjectLib
{
    importlib("stdole2.tlb");
    interface IDocHostUIHandlerDispatch;
    interface IAxWinAmbientDispatchEx;
    [
        uuid(3EDA27DF-2642-43DE-A387-5B420CDA9AEF),
        helpstring("BodyGuideClass Class")
    ]
    coclass BodyGuideClass
    {
        [default] interface IBodyGuideClass;
    };
};
};

```

Appendix B: BodyGuideSmartDevice Code

B.1: CPP File

```
#include "stdafx.h"
#include "fstream"
#include <atlltime.h>
#include "BodyGuideClassClassproxy.h"
using namespace std;
STDMETHODIMP CBodyGuideClassClassproxy :: CalculateBMI(int iWeight, int
iHeight, double* piResult)
{
    CTime t = CTime::GetCurrentTime();
    fstream file("c:\\temp.txt", ios::app);
    file<<t.GetHour()<<":"<<t.GetMinute()<<":"<<t.GetSecond()<<"-
    "<<"CalculateBMI() operation started..."<<endl;
    int sHour = t.GetHour();
    int sMinute = t.GetMinute();
    int sSecond = t.GetSecond();
    long sTime = (sHour * 60 * 60) + (sMinute * 60) + sSecond;
    HRESULT hr = proxy_obj->CalculateBMI(iWeight, iHeight, piResult);
    file<<t.GetHour()<<":"<<t.GetMinute()<<":"<<t.GetSecond()<<"-
    ""CalculateBMI() operation finished..."<<endl;
    file<<"Function Arguments = int iWeight : "<<iWeight<<endl;
    file<<"Function Arguments = int iHeight : "<<iHeight<<endl;
    file<<"Function Arguments = double* piResult : "<<piResult<<endl;
    file.close();
    return hr;
}
STDMETHODIMP CBodyGuideClassClassproxy :: CalculateBMR(int iWeight, int
iHeight, int iAge, int iGender, float* piResult)
{
    CTime t = CTime::GetCurrentTime();
    fstream file("c:\\temp.txt", ios::app);
    file<<t.GetHour()<<":"<<t.GetMinute()<<":"<<t.GetSecond()<<"-
    "<<"CalculateBMR() operation started..."<<endl;
    int sHour = t.GetHour();
    int sMinute = t.GetMinute();
    int sSecond = t.GetSecond();
    long sTime = (sHour * 60 * 60) + (sMinute * 60) + sSecond;
    HRESULT hr = proxy_obj->CalculateBMR(iWeight, iHeight, iAge, iGender,
piResult);
    file<<t.GetHour()<<":"<<t.GetMinute()<<":"<<t.GetSecond()<<"-
    ""CalculateBMR() operation finished..."<<endl;
    file<<"Function Arguments = int iWeight : "<<iWeight<<endl;
    file<<"Function Arguments = int iHeight : "<<iHeight<<endl;
    file<<"Function Arguments = int iAge : "<<iAge<<endl;
    file<<"Function Arguments = int iGender : "<<iGender<<endl;
    file<<"Function Arguments = float* piResult : "<<piResult<<endl;
    file.close();
    return hr;
}
STDMETHODIMP CBodyGuideClassClassproxy :: CalculateBodyFat(int iWeight, int
iWaist, int iWrist, int iHips, int iForearm, int iGender, float* piResult)
{
    return proxy_obj->CalculateBodyFat(iWeight, iWaist, iWrist, iHips,
iForearm, iGender, piResult);
}
```

```

STDMETHODIMP CBodyGuideClassClassproxy :: CalculateTargetWeight(int
iWeight, float fBodyFat, int iGender, int* piResult)
{
return proxy_obj->CalculateTargetWeight(iWeight, fBodyFat, iGender,
piResult);
}

```

B2: Header File:

```

#pragma once
#ifdef STANDARDSHELL_UI_MODEL
#include "resource.h"
#endif
#ifdef POCKETPC2003_UI_MODEL
#include "resourceppc.h"
#endif
#ifdef SMARTPHONE2003_UI_MODEL
#include "resourcesp.h"
#endif
#ifdef AYGSHELL_UI_MODEL
#include "resourceayg.h"
#endif
#import"C:\TestApplication\BodyGuide\BodyGuideCOMObject\Windows Mobile 5.0
Pocket PC SDK (ARMV4I)\Debug\BodyGuideCOMObject.tlb" no_namespace
raw_interfaces_only
class ATL_NO_VTABLE CBodyGuideClassClassproxy:
public CComObjectRootEx<CComMultiThreadModel>,
public
CComCoClass<CBodyGuideClassClassproxy,&CLSID_BodyGuideClassClassproxy>,
public IBodyGuideClass
{
public: CBodyGuideClassClassproxy() {}
#ifdef _CE_DCOM
DECLARE_REGISTRY_RESOURCEID(IDR_BODYGUIDECLASSPROXY)
#endif
BEGIN_COM_MAP(CBodyGuideClassClassproxy)
COM_INTERFACE_ENTRY(IBodyGuideClass)
END_COM_MAP()
DECLARE_PROTECT_FINAL_CONSTRUCT()
HRESULT FinalConstruct()
{
HRESULT hr = CoCreateInstance(__uuidof(BodyGuideClass), NULL,
CLSCTX_INPROC_SERVER, __uuidof(IBodyGuideClass), (void**) &proxy_obj);
if(FAILED(hr))
MessageBox(NULL, L"CoCreateInstance failed",L"",0);
else
MessageBox(NULL, L"CoCreateInstance succeeded",L"",0);
return hr;
}
void FinalRelease()
{
proxy_obj->Release();
}
public:
STDMETHOD (CalculateBMI)(int iWeight, int iHeight, double* piResult);
STDMETHOD (CalculateBMR)(int iWeight, int iHeight, int iAge, int iGender,
float* piResult);
STDMETHOD (CalculateBodyFat)(int iWeight, int iWaist, int iWrist, int
iHips, int iForearm, int iGender, float* piResult);

```

```

STDMETHOD (CalculateTargetWeight)(int iWeight, float fBodyFat, int iGender,
int* piResult);
private:
IBodyGuideClass *proxy_obj;
};OBJECT_ENTRY_AUTO(__uuidof(BodyGuideClassClassproxy), CBodyGuideClassClass
proxy)

```

B3: IDL File

```

// BodyGuideSmartDevice.idl : IDL source for BodyGuideSmartDevice
//

// This file will be processed by the MIDL tool to
// produce the type library (BodyGuideSmartDevice.tlb) and marshalling
// code.

import "oaidl.idl";
import "ocidl.idl";

[
    uuid(589ABE71-8262-4CCB-A21F-9AC02DAED61B),
    version(1.0),
    helpstring("BodyGuideSmartDevice 1.0 Type Library")
]
library BodyGuideSmartDeviceLib
{
    importlib("stdole2.tlb");
    importlib ("C:\\TestApplication\\BodyGuide\\BodyGuideCOMObject\\Windows Mobile
5.0 Pocket PC SDK (ARMV4I)\\Debug\\BodyGuideCOMObject.tlb");
    interface IDocHostUIHandlerDispatch;

    [
        uuid(5A1ACD68-36A2-463e-BBF2-E056B9F7233A),
        helpstring("BodyGuide Class")
    ]
    coclass BodyGuideClassproxy
    {
        [default] interface IBodyGuideClass;
    };
    interface IAxWinAmbientDispatchEx;
};

```